Active Flash:
Out-of-core Data Analytics on Flash Storage

Simona Boboila[1], Youngjae Kim[2], Sudharshan S. Vazhkudai[2], Peter Desnoyers[1], and Galen M. Shipman[2]

[1]Northeastern University, [2]Oak Ridge National Laboratory
[1]{simona, pjd}@ccs.neu.edu, [2]{kimy1, vazhkudaiss, gshipman}@ornl.gov

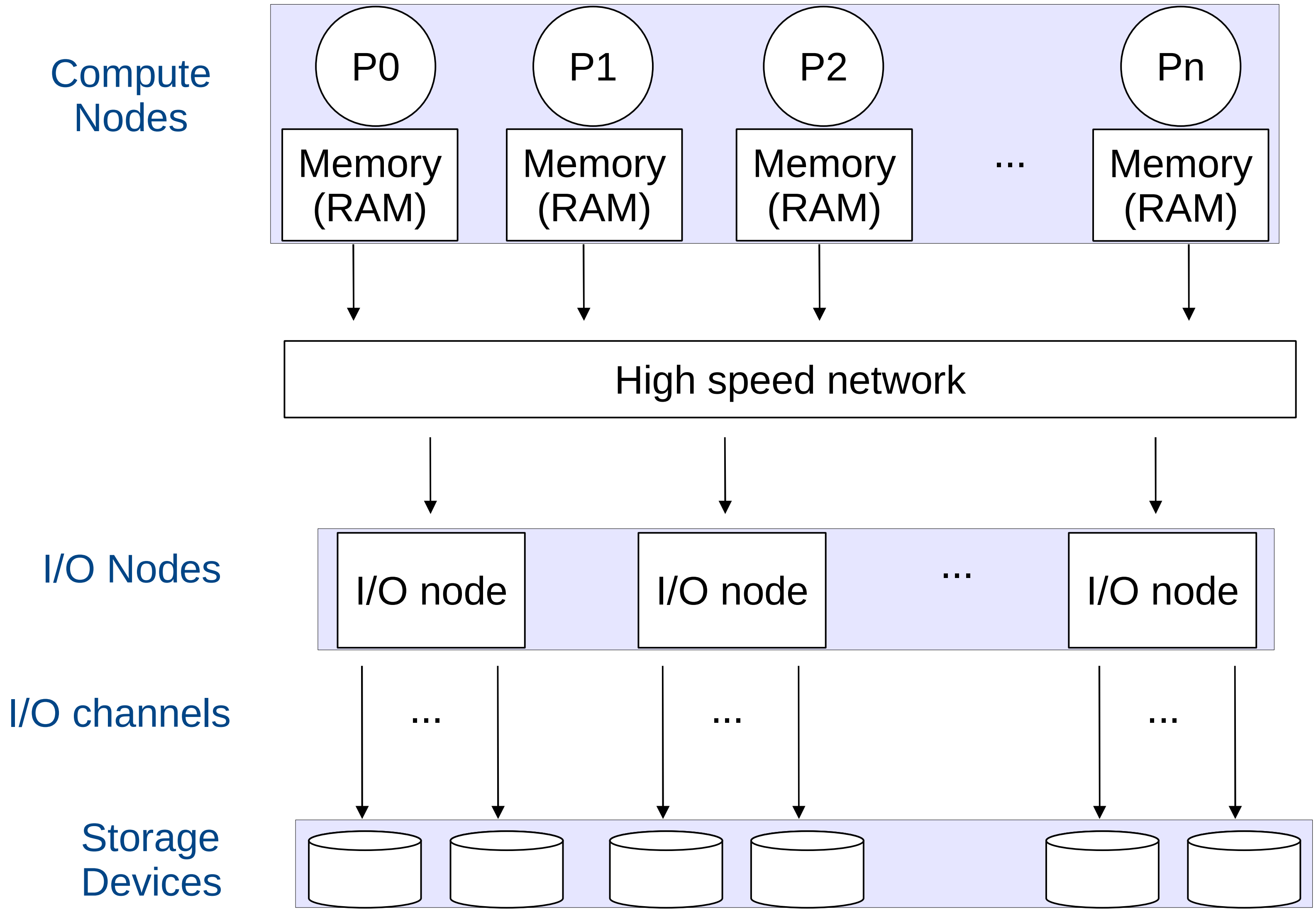
Context

High-performance computing (HPC):

- Computation-intensive tasks (scientific applications):
  - quantum physics, climate research, molecular modeling, physical simulations, etc.
- Large amounts of data generated and sent to persistent storage for later post-processing (data analysis).
- Multiple I/O rounds to send data back and forth between storage and compute nodes for post-processing.

Context - HPC

Jaguar



224,256 cores (1.75 petaflops)

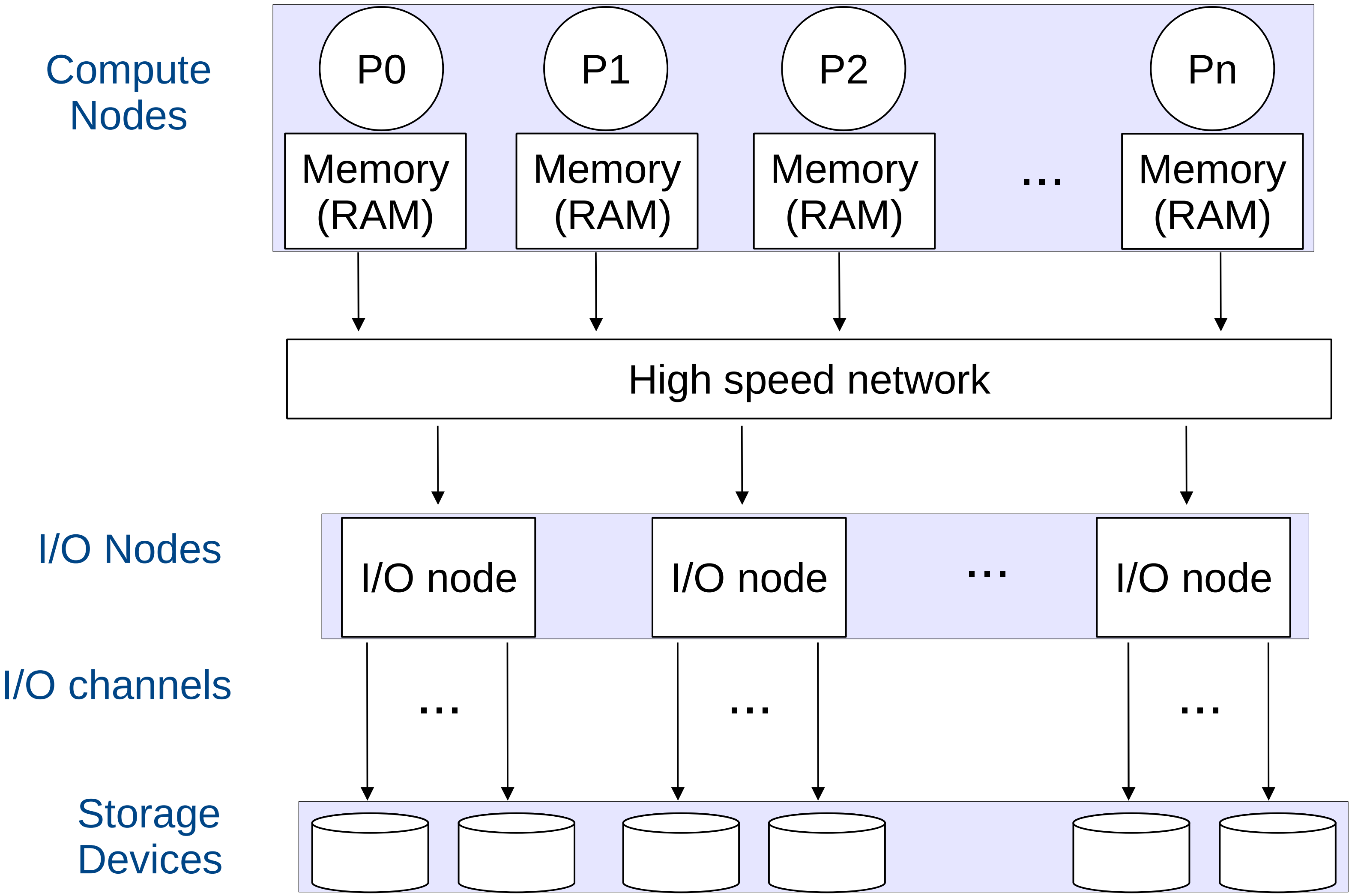
360 TB total RAM = 2 GB/core

I/O Bandwidth: 240 GB/s total = 1 MB/s per core

10 PB total storage

Context and Motivation

Jaguar



224,256 cores
(1.75 petaflops)

360 TB total RAM
= 2 GB/core

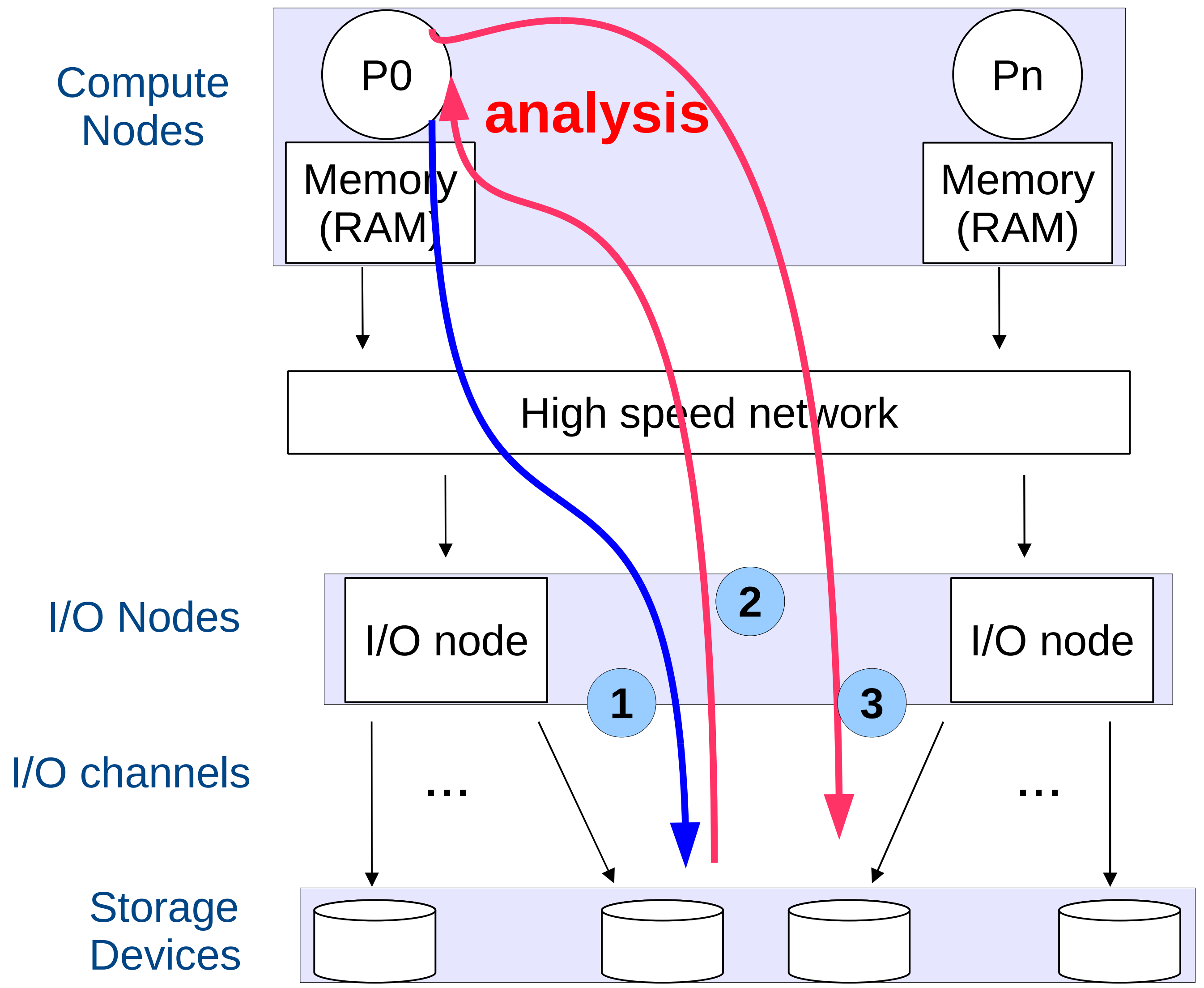
I/O Bandwidth:
240 GB/s total
= **1 MB/s per core**

10 PB total storage

Issues:

- **I/O bandwidth has failed to keep up with computation**
- **High power consumption (5 - 10 megawatts)**

Context and Motivation

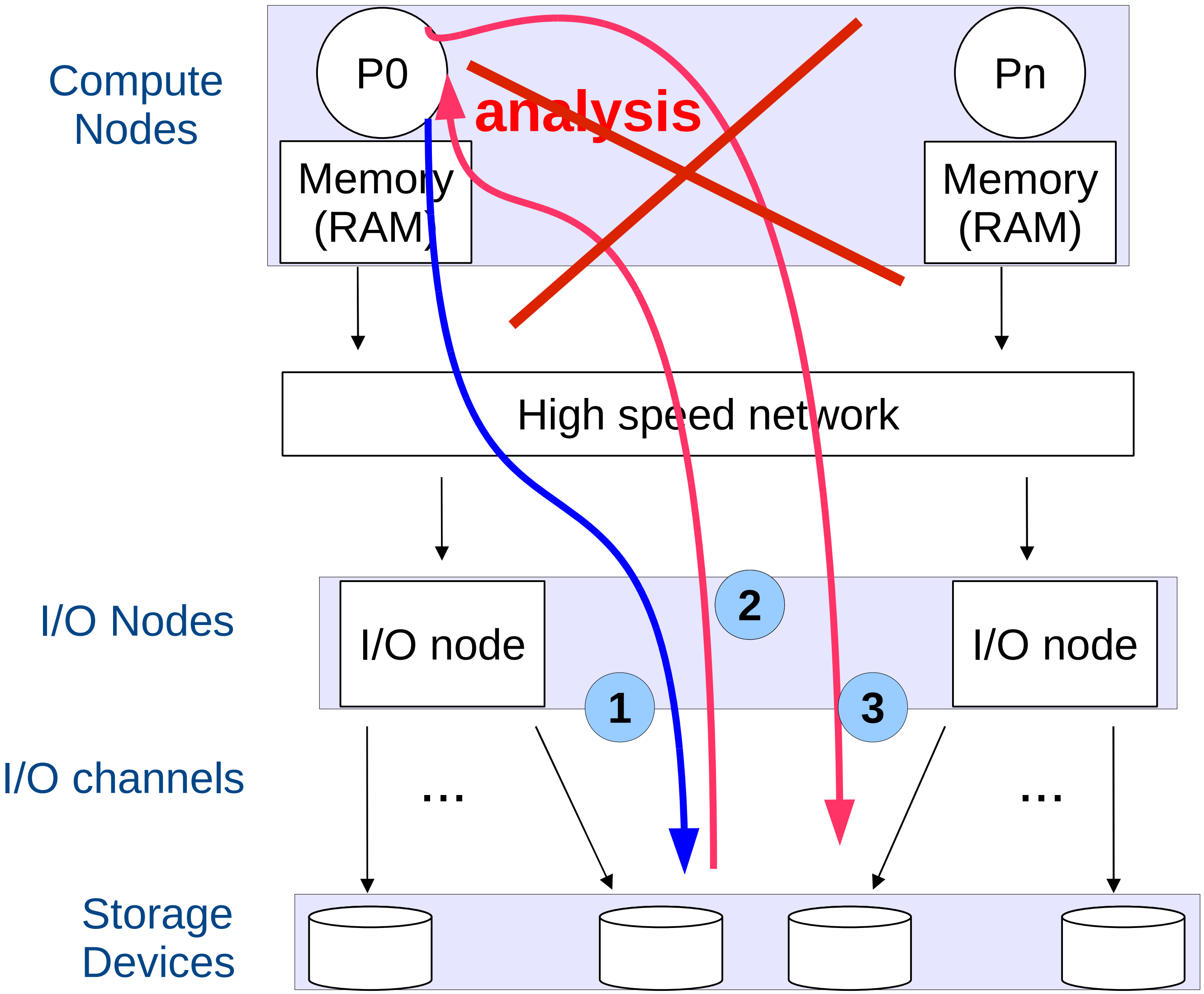


1. Generate simulation data (scientific application)
2. Read data for analysis (post-processing)
3. Write results of analysis (post-processing)

Multiple I/O rounds!

- The data transfer cost is even higher if the scientific application and the post-processing application run on different clusters!

Active Flash

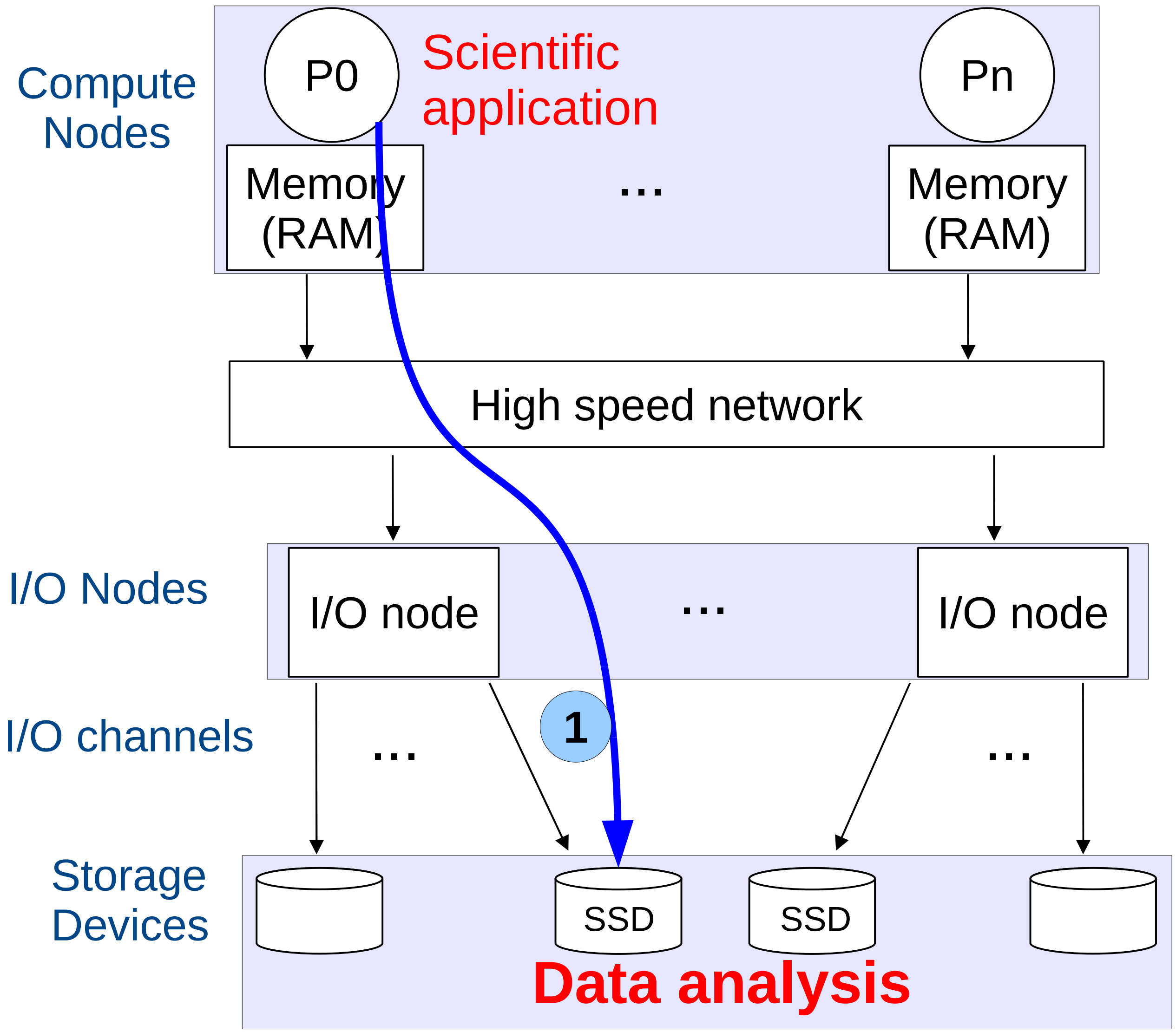


1 Generate simulation data (scientific application)

2 Read data for analysis (post-processing)

3 Write results of analysis (post-processing)

Active Flash – Scenario 1



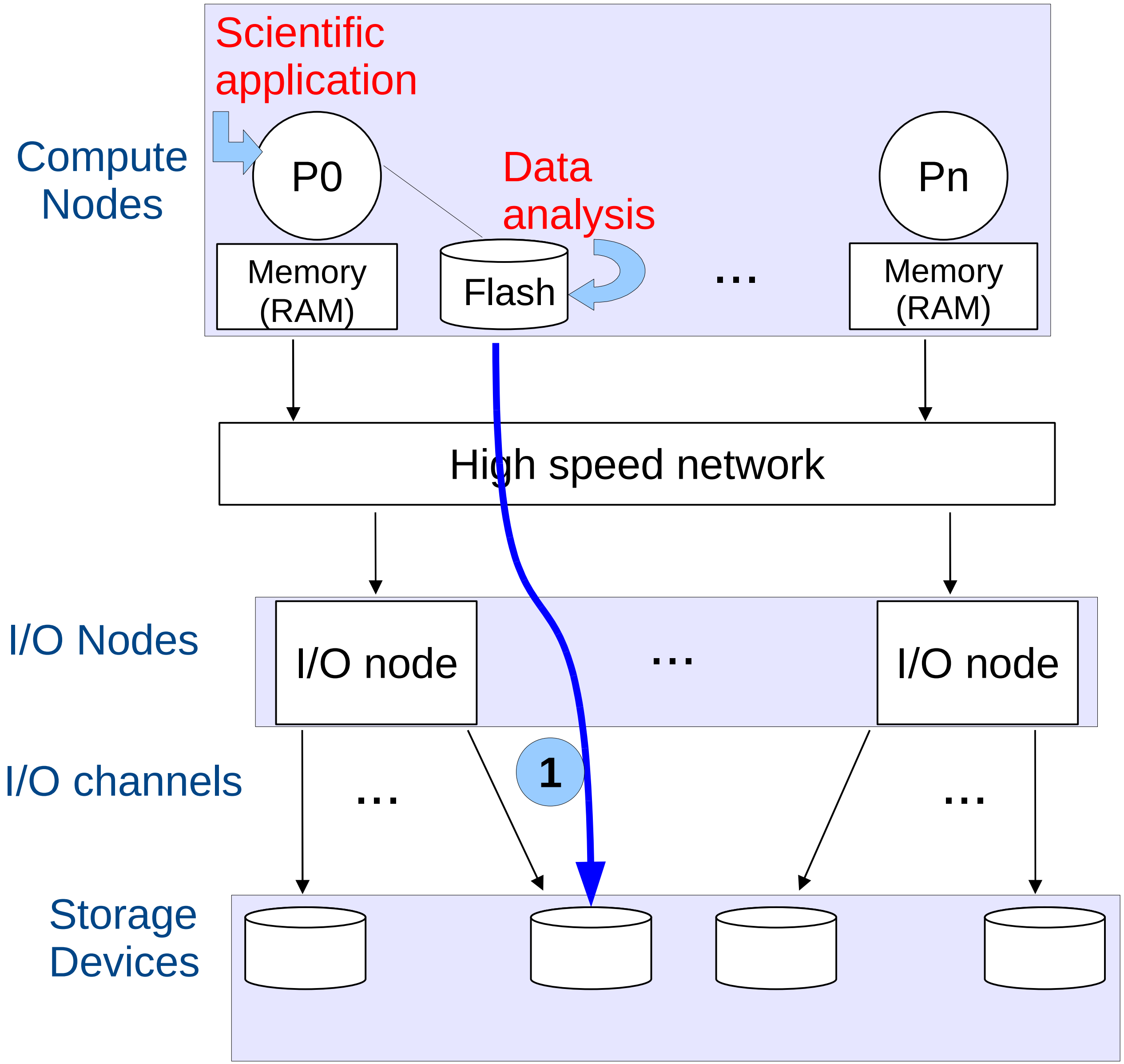
Active Flash:
Run data analysis (partially or entirely) on the embedded SSD storage controller.

Advantages:

- Reduce I/O traffic
- Save energy
- Run analysis in parallel with the scientific application

1 Generate simulation data (scientific application)

Active Flash - Scenario 2



Active Flash:
Run data analysis (partially or entirely) on the embedded Flash storage controller.

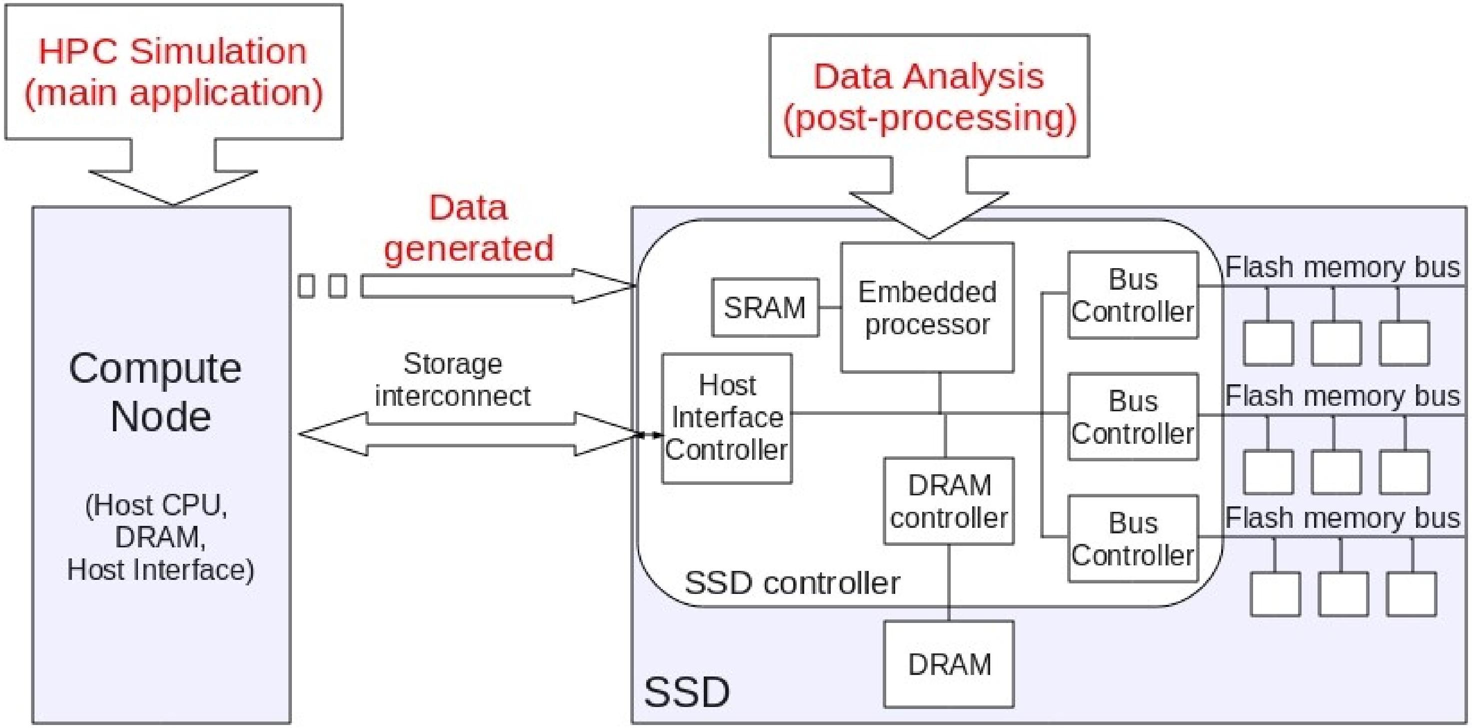
- Compute node-local flash and analysis therein

Advantages:

- Reduce data coming out of the compute node itself
- Save energy
- Run analysis in parallel with the scientific application

1 Generate post-processing results

Active Flash architecture and feasibility



Feasibility aspects:

- High I/O bandwidth of SSDs: 100-500 MB/s
- Availability of idle times in workloads due to low I/O latencies of SSDs and HPC workload burstiness.
- High-performance embedded processors
  - SSD controllers: 80-800 MHz per core; 1-4 cores;
  - “mobile-class” processors: e.g. ARM Cortex-A9 at 2000 MHz per core; 1-4 cores

Active Flash versus Active Disks

- Active storage: run computations on the embedded storage controller:
  - inside the Hard Drive (HDD) - Active Disks (in the '90s)
  - inside the Solid-State Drive (SSD) - Active Flash
- Active Flash overcomes the architectural limitations of Active Disks:
  - Higher I/O bandwidth of SSDs
  - Lower I/O latencies of SSDs
  - Higher-performance embedded processors in SSDs

Contributions

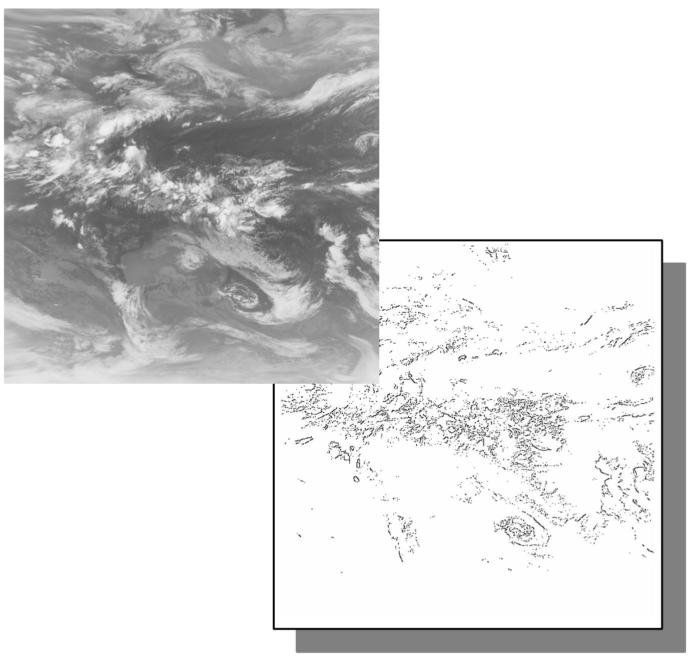
Active Flash: Run data analysis (partially or entirely) on the SSD storage controller

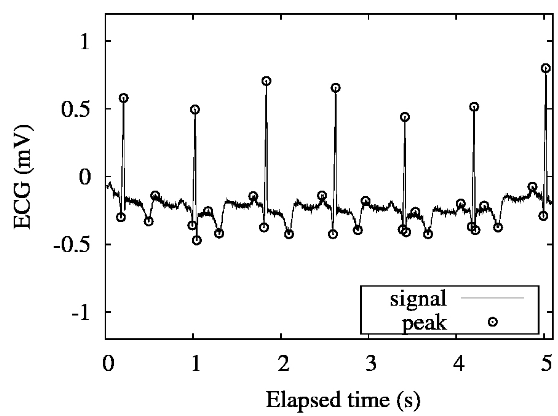
- Evaluation on realistic data analysis applications
- Exploration of performance-energy trade-offs
- Proposal and implementation of possible scheduling policies, and evaluation of compute-IO trade-offs by simulation

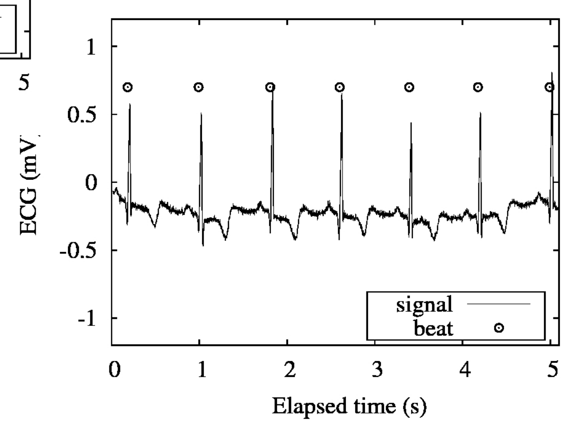
Active Flash evaluation on realistic data analysis applications

Data analysis applications:

- data compression (LZO) on scientific data (NetCDF and text)
- edge detection on atmospheric data (PGM images)
- local extrema detection on medical data (text)
- heartbeat detection on medical data (binary)







Active Flash evaluation on realistic data analysis applications

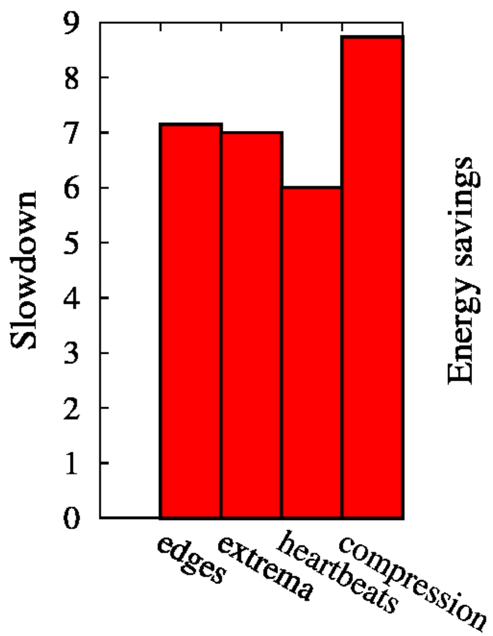
Experimental platforms:

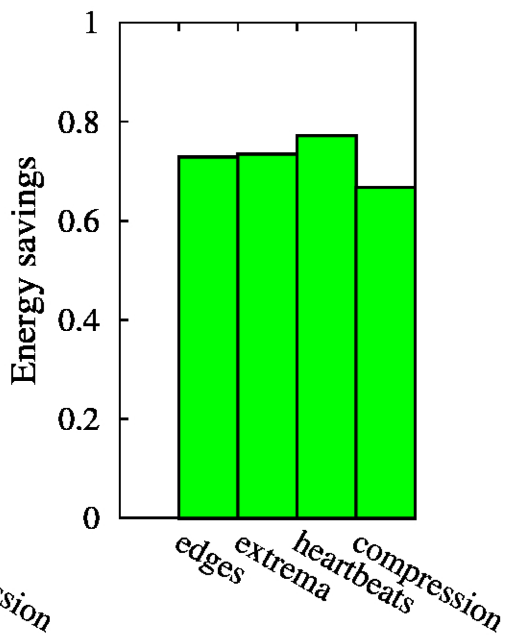
- Pandaboard development system:
  - dual-core 1 GHz ARM Cortex-A9 (~ SSD Controller),
  - 1 GB of DDR2 SDRAM,
  - Linux kernel 3.0
- Host machine:
  - Intel Core 2 Quad CPU Q6600 at 2.4 GHz (Host CPU),
  - 4 GB of DDR2 SDRAM,
  - Linux kernel 2.6.32
- Measured computation phase (no I/O) on a single core with no competing processes

Active Flash evaluation on realistic data analysis applications

Performance–Energy tradeoffs of data analysis running on:
SSD controller compared to host CPU

Slowdown versus energy savings





Slowdown: $S = \frac{t_{ssd}}{t_{host}}$

Energy savings:

$$E = t \times P$$

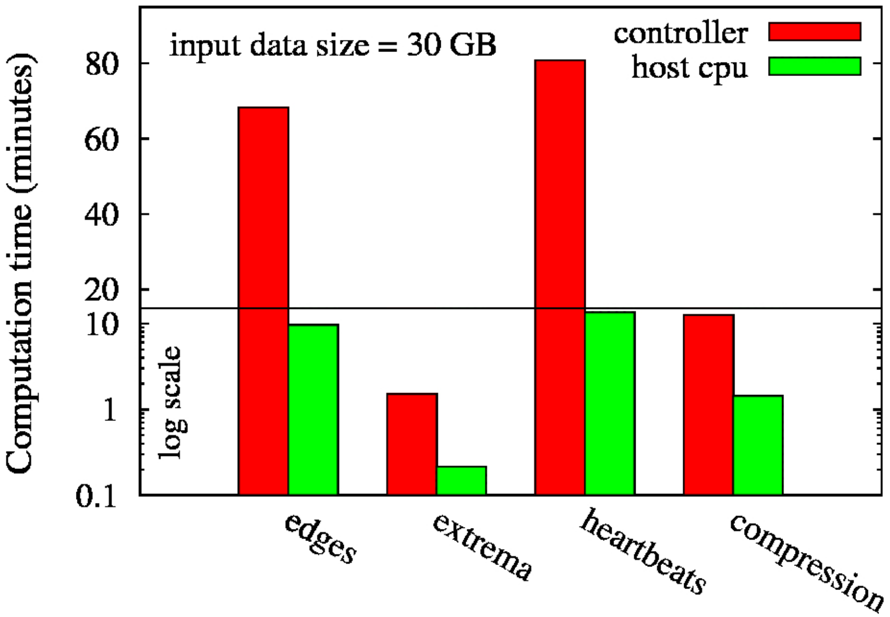
$$\Delta P = P_{load} - P_{idle}$$

$$\Delta E = 1 - \frac{\Delta E_{ssd}}{\Delta E_{host}} = 1 - \frac{t_{ssd}}{t_{host}} \frac{\Delta P_{ssd}}{\Delta P_{host}}$$

- Running data analysis on the SSD controller instead of the host CPU results in about 7x slowdown and 70% energy savings.

Active Flash evaluation on realistic data analysis applications

Computation time on the SSD controller for
30 GB input data (*)



(*) Gordon system at San Diego Supercomputing Center has 64 GB DRAM per node, 1024 16-core nodes.

- Consider periodic checkpoints of half the node's DRAM memory.

Scheduling data analysis on flash

- **On-the-fly data analysis** – while data is still in the SSD-resident DRAM
  - Less internal I/O traffic inside the SSD
  - Data analysis needs to keep up with the data generation rate from the host CPU (b/c DRAM size is limited)
- **Idle-time data analysis** – data written to SSD, analyzed later during idle times
  - Can handle higher data generation rates from the host CPU (b/c data analysis scheduled with low priority, only in idle times)
- **Idle-time data analysis with GC management** – when no data to process, schedule GC during idle times
  - The early GC can increase write amplification (due to less stale data), but with no impact on performance (since it happens in idle times)

Scheduling data analysis on flash

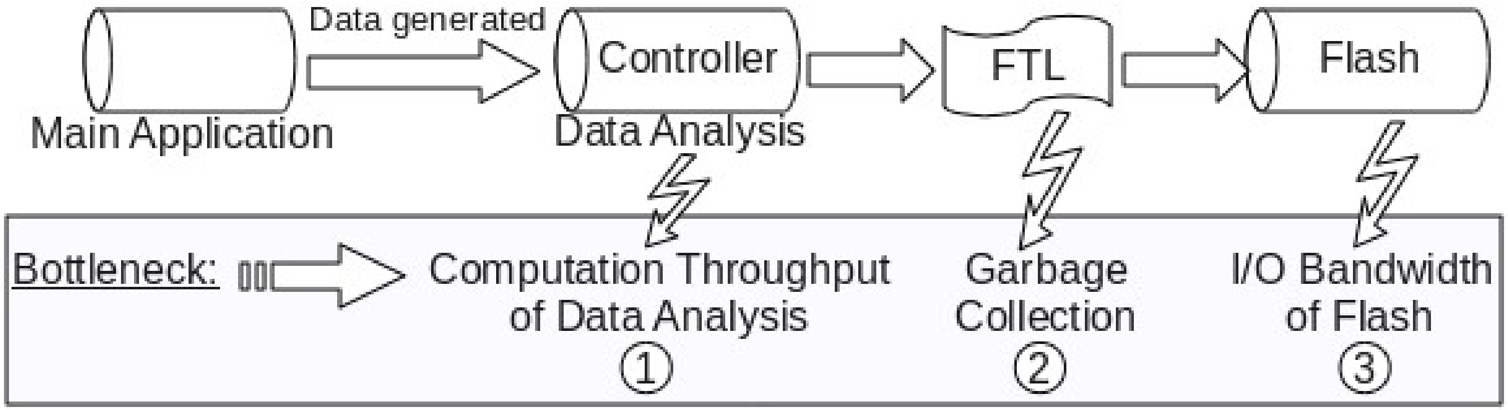
- Scheduling policies were implemented in the Microsoft Research SSD simulator.

Data analysis-related simulator parameters

Parameter	Value
Computation time per page of input	application-specific
Data reduction	application-specific
GC-idle threshold (fraction of reserved space)	0.9

Scheduling data analysis on flash

Bottlenecks in Active Flash



① → computation bound data analysis

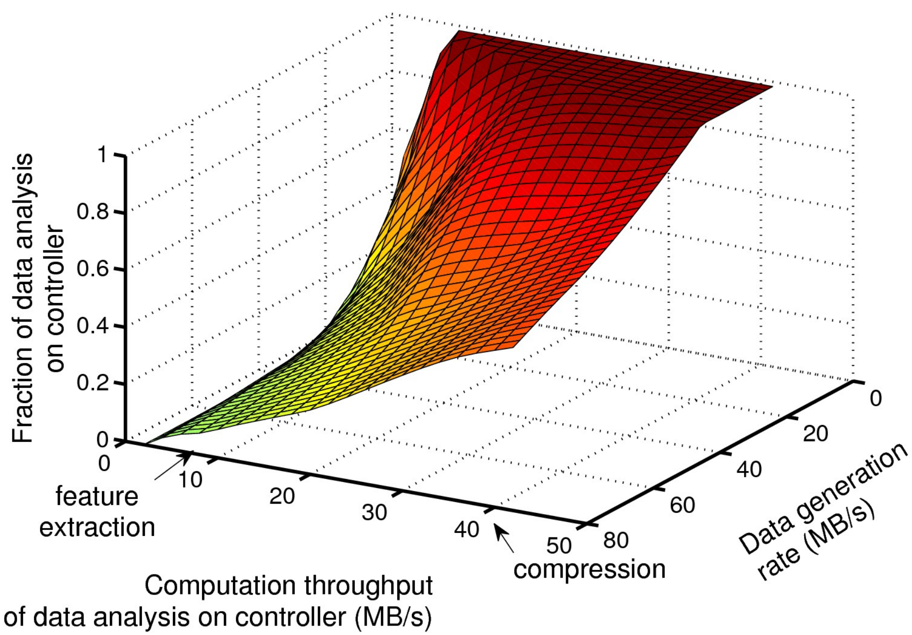
③ → I/O bound data analysis

- Scheduling policies were evaluated in relation with potential bottlenecks.

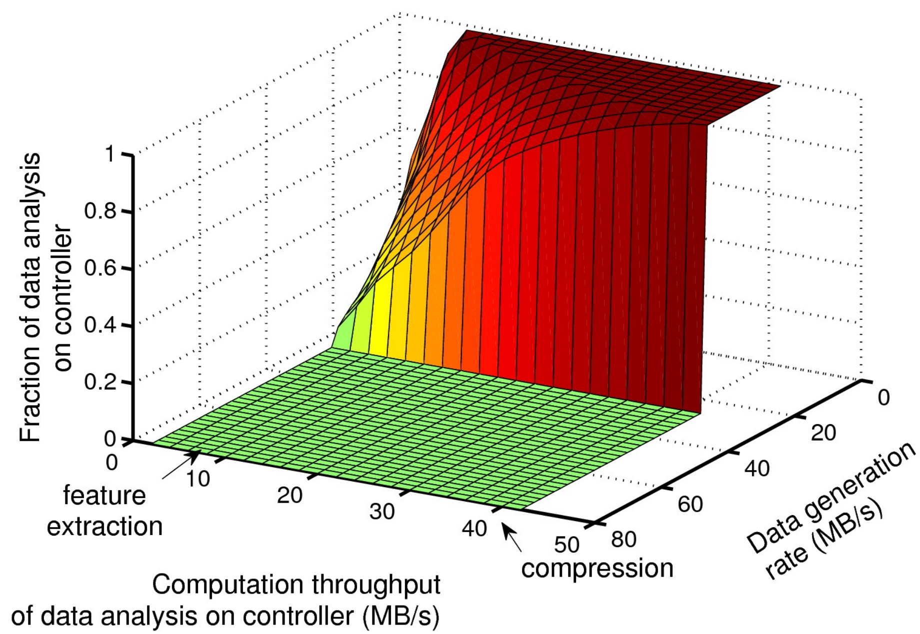
Scheduling data analysis on flash

- Data analysis on the SSD controller runs in parallel with the scientific application that generates data on the host CPU.
- What fraction of data analysis is completed on the SSD controller during the execution of the scientific application?

Idle-time computation bound data analysis

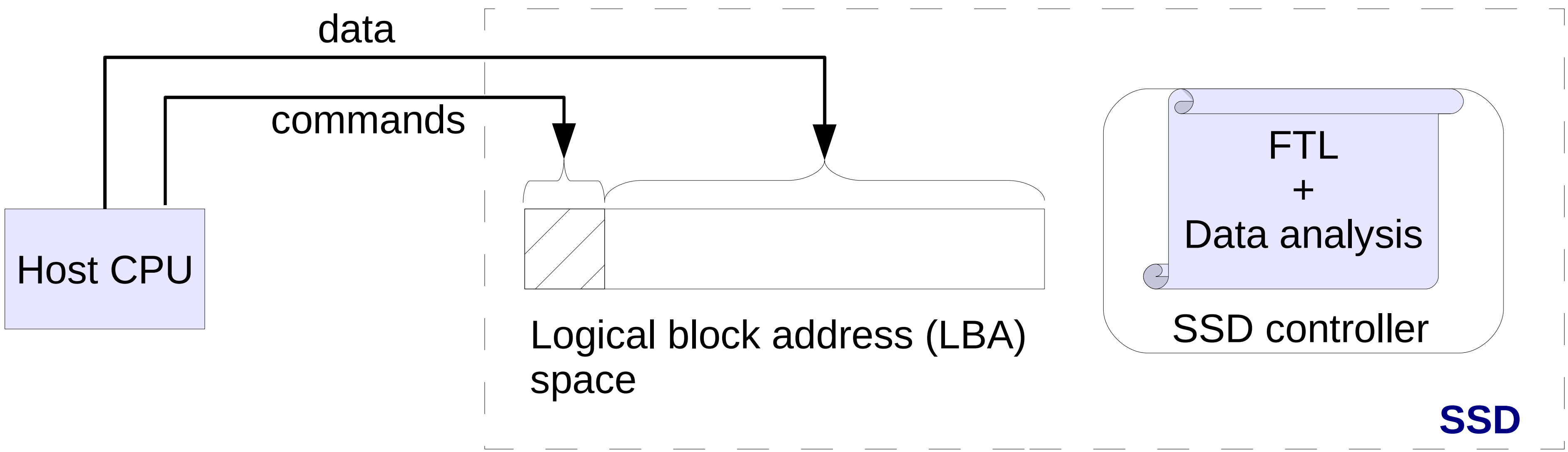


(a) “free” (no GC)



(b) “full” (intensive GC)

Implementation aspects



Real-world implementation entails:

- Extension of current SSD interfaces:
  - Host-controller communication using a dedicated flash memory region for commands
  - Commands encode: analysis type, LBA of input data
- Implementation of data analysis inside the SSD:
  - Algorithms for analysis of scientific data co-located with Flash Translation Layer (FTL) logic

Northeastern

OAK RIDGE
National Laboratory

Questions?